V I S T A

Patch Module 2.5

User Manual: Secondary Developers

September 2014

prepared for

OSEHRA

Open Source Electronic Health Record Alliance

by

©Copyright 2014 by VISTA Expertise Network. Licensed under Creative Commons Attribution-ShareAlike 4.0 International. Details are available at http://creativecommons.org/licenses/by-sa/4.0/

Revision History

Date	Description	Language	Authors
September 2014	OSEHRA Version 2.5 release	English (US)	Kathy Ice and Frederick D.S. Marshall

Contents

Orientation....1
Introduction....2
Chapter 1: Overview....3
The Patch Process....4
Chapter 2: Evaluating Patches....7
Extended Display of a Patch....7
Edit a Patch: In Review....7
Chapter 3: Secondary Development....11
Add a Patch....11
Edit a Patch: Secondary Development....12
Copy a Patch into a New Patch....18
Create a PackMan Message....18
Appendix A: The Lineage of VistA....19
Glossary....20

Orientation

This manual is one of four user manuals for version 2.5 of the Patch Module. There is a different user manual for each essential role associated with the Patch Module; this manual is for secondary developers.

If you are the initiator of a patch, or patch stream, then you are a primary developer. If you are taking patches issued by another organization, altering them to work with your dialect of VistA, and re-releasing them to your customers, you are a secondary developer.

Primary developers have their own Patch Module user manual, which can be found at www.osehra.org along with the rest of the Patch Module documentation suite.

Verifiers and patch subscribers also have their own Patch Module user manuals,which can be found at www.osehra.org along with the rest of the Patch Module documentation suite.

The Patch Module documentation suite:

Release Notes
Value Proposition
Installation Guide
User Manual: Primary Developers
User Manual: Secondary Developers
User Manual: Verifiers
User Manual: Patch Subscribers
Technical Manual
Security and Privacy Manual

Introduction

The Patch Module, as the name implies, is a software package that allows users and developers to create, revise, distribute, review, and receive software patches and updates for VistA. Options are provided for systematic entry, revision, and review of patches by developers, review and release of patches by verifiers, and display and distribution of the released patches to the users.

But before we get to talking about all the things we can do with patches, it's probably best to take a moment and make sure we're all on the same page about what a patch is, and what it does. Patches began as a way of fixing problems in an active VistA system, and many patches released today are simple bug fixes. However, most patches include more than fixes. Developers found that patches were the easiest way to add the enhancements and new features that their users wanted without taking the system offline or releasing a new version. Today, patches are the primary method for making updates and improvements to VistA.

A patch, then, can include bug fixes, upgrades, enhancements, new features, or all of the above. Its main feature is that it can be installed on an active, running VistA system with minimal disruption.

Patches are created in KIDS (the Kernel Installation and Distribution System), but are packaged and distributed via the Patch Module. Until relatively recently, only the developers at the Department of Veterans Affairs (VA) had access to the Patch Module. With this release, the Patch Module becomes more widely available, and more developers have the opportunity to create and release patches.

Chapter 1: Overview

Medicine is in a constant state of change. This means that our customers, the medical professionals at the facilities we support, are in a constant state of needing new things. The best way to deliver those changes in a timely manner is through patches. Patches are small, self-contained, and can be installed without disrupting ongoing operations.

The patching process begins with the primary developers. They code the required change into VistA, whether it be a bug fix, a new menu option, an enhancement to an existing feature, or some other change. Once the new code has been tested and finalized, it is prepared for distribution using KIDS, the Kernel Installation and Distribution System. The primary developers then use the Patch Module to create and distribute the actual patch to their customers.

The patch created by the primary developers is compatible with their dialect of VistA. For example, a patch created by VA developers is compatible with VA VistA. It may not be compatible with other dialects of VistA—even FOIA VistA—because VA VistA includes proprietary components. This brings us to the secondary developers.

The process of secondary development for patches begins with a patch reviewer. The patch reviewer assesses the patch as it was issued by the primary developers, and makes a determination for their dialect of VistA. The determination can be one of three results:

1. The patch can be installed as distributed (no changes necessary).
2. The patch will not be installed in our systems.
3. The patch needs modifications before it can be installed.

If the patch reviewer determines that modifications are needed, the secondary developers go to work, making the necessary changes to make the patch compatible with their dialect of VistA. Once the changes are

complete and tested, the secondary developers use KIDS and the Patch Module to create and distribute their modified patch to their customers. Because each dialect of VistA is separate and distinct, each needs its own team of secondary developers to evaluate and modify patches.

Now that we've seen how the overall process works, let's take a closer look at the specific steps performed by the secondary developers.

The Patch Process

Step 0: Development Environment

The new code that will become the patch should be developed in a development environment, not a production environment. Most developers are aware of this rule, and most of them violate it from time to time. However, that doesn't make it a good idea. Develop the code in a development environment, not a production environment. Even when you're in a hurry.

Step 1: The Primary Patch

Secondary development begins when a primary-development patch is released. The patch is loaded into the secondary-development environment's Forum system, which automatically makes a copy of the patch and numberspaces it for secondary development.

Step 2: Patch Review

The developer assigned to review the patch performs an assessment to determine whether the released patch will be installed in their dialect, and if so, whether any changes will be required.

If the patch reviewer determines that the patch will not be installed at all, or that it can be installed "as is," the secondary development process is essentially complete. However, if the patch reviewer determines that the patch requires modification, the secondary development process is just beginning.

Step 3: Secondary Development

Next, the secondary developers make the required changes to make the patch compatible with their dialect of VistA. Once the code is completed, the secondary developers use KIDS to send an email to OSEHRA Forum (XXX@Q-PATCH.OSEHRA.ORG). This makes the code available to the Patch Module for inclusion in the patch.

Step 4: Secondary Testing

Before it can be completed and sent to the verifiers, the new patch needs to be tested. There are three sub-phases to this step:

Internal Testing
Alpha Testing
Beta Testing

In a fast-paced environment, it can be tempting to skip one of these phases, but that is not a good idea. VistA customers depend on these patches working as intended, without unforeseen problems, and the only way to ensure that is with rigorous testing.

Step 5: Secondary Patch Completion

Once testing is complete, the patch completer performs a final review of the patch and the code. Once the patch completer is satisfied that the patch is ready for distribution, he or she uses the Patch Module to set the patch's

status to “Sec Completion.” This automatically triggers the secondary verification process.

Step 6: Corrections (If Needed)

If the verifier finds any problems with the patch, the secondary development team will need to address the problems. This may involve additional coding and testing before the patch can be re-completed.

Chapter 2: Evaluating Patches

Secondary development begins with the Forum system manager, who imports the released patch into the secondary development environment. This process automatically creates a two copies of the patch: one for FOIA release, and the other numberspaced for secondary development. Both copies are given a default status of "In Review." The patch reviewer can then assess the patch, looking for any areas where it may conflict with secondary development already installed in the environment. For this reason, it is preferable that the patch reviewer be a person very familiar with the package being patched, and how that package has been modified for this dialect of VistA.

Extended Display of a Patch

If you are a patch reviewer, Extended Display of a Patch is probably where you want to start. This option allows you to see all the available information about the patch, including internal comments. If you want to send yourself a copy of the patch to install, you can use the Edit a Patch option to do so.

Edit a Patch: In Review

The Edit a Patch option is accessed from the developer's menu. For this example, we're going to assume we're reviewing a patch from the imaginary Exampleman package—the same patch, in fact, that we "created" in the Primary Developer Manual

```
Select Developer's Menu Option: EDIT a Patch

Select PATCH: ZZ*22.2*10001
STATUS OF PATCH: IN REVIEW//
```

When editing a patch that has a status of "In Review," the prompts you see are a little different. You will not be prompted for a patch description, priority, category, or anything like that. If you are reviewing a patch, the assumption is that you are not making changes. Editing a patch that is in review is therefore limited to changing the status and adding comments.

```
STATUS OF PATCH: IN REVIEW//
IN REVIEW DESCRIPTION:
  THERE ARE NO LINES!
  Edit? NO//
```

If you choose to add an "In Review" description, it will be handled as an internal comment, visible to other developers but not visible in a standard patch display, and not sent as part of the patch text. Use this field to keep your team informed of the status of your review, and any issues you encounter with the patch.

To send a copy of the patch for installation—to yourself and to anyone who may be assisting you with the process—enter "In Review" as the status, rather than accepting the default.

```
STATUS OF PATCH: IN REVIEW// IN REVIEW

   Option to create a Patch message to send to test sites.
   TEST v1   will be added to the Patch message subject.

You may change the TEST v[#] if necessary.:  (1-99): 1//
```

When "In Review" is re-entered as the status, we get the option to send a Patch message to test sites. We can change the version number if we like, or accept the default. We are then prompted for the patch recipients. Once the recipients have been entered, the Patch Module echoes back some version-control information.

```
Please add recipients for 'ZZ*22.2*10001' test message
Forward mail to: REVIEWER,RICK
And Forward to:
message number [#174]
NOTE: A message has been sent to the Test Site Users for
distribution.
```

```
      Please use mailman if you need to forward this message
again.


Exporting Patch  ZZ*22.2*10001
65/KIDComponents/
Wrote Build.zwr
Wrote Package.zwr
Wrote KernelFMVersion.zwr
Wrote EnvironmentCheck.zwr
Wrote PostInstall.zwr
Wrote RequiredBuild.zwr
Wrote InstallQuestions.zwr
Exporting these routines to Routines/
IN REVIEW DESCRIPTION:
  THERE ARE NO LINES!
  Edit? NO//
```

One of the improvements to version 2.5 of the Patch Module is compatibility with standard version-control software such as Git. The information displayed in this transaction can be imported into Git to ensure that your system remains properly version-controlled. It is highly recommended that all sites running VistA begin to move toward standardized version control.

Once your review of the patch is complete, you will again use the "Edit A Patch" option, this time to change the status. You'll want to change the status on both copies of the patch: the FOIA version and the version for your organization.

If you have determined that the patch can be installed as is, with no further development required, change the status of both copies to "Sec Completion." This triggers the secondary verification process. Yes, we're not just going to take your word for it; we're going to check first. The verifiers will be notified, and secondary verification will begin.

If you have determined that the patch should not be installed at all, change the status of your organization's patch to "Not For Sec Release."

```
STATUS OF PATCH: IN REVIEW// NOT FOR  NOT FOR SEC RELEASE
```

```
Are you sure you want to change status to Denied Secondary
Release? No// Yes
   ...status changed to Denied Secondary Release
   . . ..................................................
NOTE: A message has been sent to the Test Site Users for
distribution.

NOTE: A bulletin has been sent to users who have viewed this
patch informing them of this Denied Secondary Release patch.

Exporting Patch ZZ*22.2*10001
67/KIDComponents/
Wrote Build.zwr
Wrote Package.zwr
Wrote KernelFMVersion.zwr
Wrote EnvironmentCheck.zwr
Wrote PostInstall.zwr
Wrote RequiredBuild.zwr
Wrote InstallQuestions.zwr
Exporting these routines to Routines/
NOT FOR SEC RELEASE DESC:
```

If you choose this option, you again see the version-control information, and then you are prompted for a description. Although anything you put in your In Review description will still be there, it is probably a good idea to put a shorter explanation in this description, explaining why this patch will not be released.

Even if your organization will not be installing the patch, the FOIA version still needs to be made available. Change the status of the FOIA patch to "Sec Completion," and be sure to include your comments about the issues you found in your review.

If you determine that the patch requires additional development before it can be released, change the status of your organization's patch to "Sec Development." This change automatically sends a bulletin to the developers, which will almost certainly be inadequate to the task of explaining exactly what needs to be done. You will probably want to communicate this information directly. And, as with a patch that won't be installed, you'll want to change the status of the FOIA version to "Sec Completion" to trigger the verification process.

Chapter 3: Secondary Development

Most of the work done by secondary developers involves choosing options from the developer's menu, which is available to any user with developer access to VistA.

```
Developer's Menu [A1AE DEVELOPER]
   DP     Display a Patch
   IN     Routine Inquire
   TS     Scan Patch for Discrepancies and Contents
          Add a Patch
          Completed/NotReleased Patches Report
          Copy a patch into a new patch
          Create a packman message
          Delete an NotReleased Patch
          Display a Completed/NotReleased Patch
          Edit a Patch
          Extended (DIQ) Display of a Patch
          Forward a Completed/NotReleased Patch Message
          Package Management ...
          Package Menu ...
          Routines that overlap in patches
          Show a Patch's Relationships
          Under Development Patches Report

Select Developer's Menu Option:
```

Add a Patch

If you are doing secondary development, you should no longer be using the Add a Patch option; starting with version 2.5 of the Patch Module there is a different, more automated process. If you are developing a brand-new patch, one that is not derived from a release patch, then you are doing primary development. Consult the Patch Module 2.5 User Manual for Primary Developers.

Edit a Patch: Secondary Development

Editing a patch with a status of "Sec Development" is similar to the process for editing an "Under Development" patch in primary development.

```
Editing Patch: ZZ*22.2*10001

PATCH SUBJECT: Fix undefined error at line 615//
HOLDING DATE:
```

First, we are prompted for the patch subject and holding date. Unless you have a compelling reason, it's probably not a good idea to change the patch subject. Distributing what is essentially the same patch, but with a different title, will probably confuse some of your users.

We are also given the opportunity to designate a holding date. This field may be useful if you are coordinating your development with other secondary developers, and you want to release your secondary patches at the same time.

```
PRIORITY: MANDATORY//
Select CATEGORY OF PATCH: ROUTINE//
```

The next prompts ask about priority and category. Generally, the priority of your secondary patch should be the same as the primary patch, unless there is a specific reason to change it.

Category of patch is a multiple field. So, while you should definitely keep the original category (or categories), it may be appropriate to add categories for your version of the patch. If you simply press Enter to accept the default, the software assumes that you have nothing to add, and moves on to the next prompt. Therefore, if you do want to add one or more categories, you'll want to type in the category, rather than accepting the default.

```
Do you want to copy lines from a message into the Patch
Description? No//
```

```
PATCH DESCRIPTION:. . .
        . . .
ZZDEMO
Done.

Associated patches:
 - ZZ*22.2*2
 - ZZ*22.2*3

  Edit? NO//Yes
```

Next, we are asked about copying lines into the Patch Message. You might want to use this option if, for example, your description will be a significant departure from the description of the original patch. In our example, however, we accepted the default answer of No. (For more information on copying lines into a Patch Message, see the Primary Developers User Manual.)

The software then shows us the existing patch description, and asks whether we want to edit it. If you want to paste in text from a text file (rather than copying lines), this is where you would do that.

When we answer "yes" to the "Edit?" prompt, we are taken to a word-processing field to make the edits, as shown below.

```
[ WRAP ]=[INSERT ]==< PATCH DESCRIPTION >[Press <PF1>H for help]
ROUTINE DELETE

All Routines? No => No

Routine: ZZDEMO
Routine:
1 routine

1 routines to DELETE, OK: NO// Y
ZZDEMO
Done.

Associated patches:
 - ZZ*22.2*2
 - ZZ*22.2*3
```

```
<=======T=======T=======T=======T=======T=======T=======T>======
                      editing MESSAGE TEXT
Do you want to copy a packman message into the Message Text?
No// yes
   (1)  Exampleman Routines              AUG 6,2014
   (2)  Exampleman Description             AUG 6,2014

Select Message to copy : 2// 1
```

Once we leave the word processor, we are asked about copying a Packman message into the Message Text. The Message Text is the part of the patch containing the actual code. You can copy the code from your KIDS email (your Packman message) into the "Message Text," which, despite its name, is actually where the Patch Module expects to find the code or software for the patch. Unlike with the Patch Description, you cannot select certain lines for the Message Text. The Patch Module assumes you want all the code, and not just some of it.

```
Select ROUTINE NAME: ZZDEMO// ?

Copy routine lines from a packman message into the description?
No//

  DESCRIPTION OF ROUTINE CHANGES:
    THERE ARE NO LINES!
    Edit? NO//
  ROUTINE CHECKSUM:
    THERE ARE NO LINES!
    Edit? NO//
Select ROUTINE NAME:
```

The next prompts have to do with descriptions for individual routines. If your patch affects more than one routine, you can describe the changes to each routine so that patch subscribers know what they'll be installing and what it will do. You can accept the description provided by the primary developers, edit it, or replace it with your own.

```
DISPLAY ROUTINE PATCH LIST: Yes//
editing comments only seen by releasers/developers
INTERNAL COMMENTS:
  THERE ARE NO LINES!
  Edit? NO//
```

```
Select PATCH RELEASE CHECK:
```

The next prompt asks about displaying the routine patch list. Choose Yes at this prompt.

We are then asked about internal comments. These are a way for developers and verifiers to communicate about the patch, and have their comments attached to the patch itself. Internal comments are not displayed in reports; the only way to see them is in "Edit a Patch" or "Extended Display of a Patch."

The next prompt asks about a patch release check. This field can be used for two related purposes: to list patches that should be verified before your patch is verified, and to list any patches that should be verified at the same time as your patch. For each patch you list, you will be asked whether the patch is required for verification. If the patch you list should be verified before your patch, choose "yes" here. If your patch and the other patch need to be verified at the same time, choose "no."

Although your list of patch release check patches will probably be similar to the one for primary development, you will need to update the list to reflect the secondary patch names. Occasionally, you may also need to include a patch that was not on the primary list.

```
SEC DEVELOPMENT DESCRIPTION:
  THERE ARE NO LINES!
  Edit? NO//
```

Next, you are asked for a Secondary Development Description. Use this field to describe the changes you made to the original patch, so that the patch completer and patch verifier know what to focus on when they complete their part of the process.

```
STATUS OF PATCH: SEC DEVELOPMENT// ?
     Choose from:
       c         COMPLETED/UNVERIFIED
       e         ENTERED IN ERROR
       u         UNDER DEVELOPMENT
```

```
      v         VERIFIED
      r         RETIRED
      x         cancel
      i2        IN REVIEW
      d2        SEC DEVELOPMENT
      s2        SEC COMPLETION
      r2        SEC RELEASE
      n2        NOT FOR SEC RELEASE
```

The next prompt asks about the patch's status. For secondary development, only the bottom five statuses listed here are actually used.

When the patch reviewer sends a patch for secondary development, the status is changed to "Sec Development" and the patch is assigned to a development team.

Once the secondary development is completed, the secondary patch completer can change the patch status to "Sec Completion." This automatically triggers the verification process.

NOTE: You can edit a patch after it is submitted for verification but the status will change back to "Sec Development."

Once the verifiers receive this bulletin they can begin the verification process of this patch. When the verifiers are ready to release the patch to the field, they can change the status of the patch to "Sec Release." When this status is assigned to the patch, a bulletin is sent to all users who have elected to be notified of patches for this package, and the patch is sent out to all patch subscribers.

Using "Edit a Patch" to Send Patches to Test Sites

Once your patch is ready for testing, you can use the Edit a Patch option to send it to your test sites. After invoking the option, go through the prompts to ensure that all the information has been entered correctly and the parameters are set the way you want them. When you get to the last

prompt, "Status of Patch," manually enter "Sec Development" rather than accepting the default, as shown here.

```
STATUS OF PATCH: SEC DEVELOPMENT// SEC DEVELOPMENT

   Option to create a Patch message to send to test sites.
   TEST v1   will be added to the Patch message subject.

You may change the TEST v[#] if necessary.:  (1-99): 1//
```

By entering "Sec Development" as the status, we trigger the process to send a patch message to test sites. The Patch Module allows us to set the version number of the test patch, with version 1 as the default. We can change the version number if we choose.

```
Please add recipients for test message
Forward mail to: TESTY,TRACY
And Forward to: tom.testworthy@testsite.org
And Forward to:
message number [#24860]
NOTE: A message has been sent to the Test Site Users for
distribution.
      Please use mailman if you need to forward this message
again.

Exporting Patch  ZZ*22.2*10001
65/KIDComponents/
Wrote Build.zwr
Wrote Package.zwr
Wrote KernelFMVersion.zwr
Wrote EnvironmentCheck.zwr
Wrote PostInstall.zwr
Wrote RequiredBuild.zwr
Wrote InstallQuestions.zwr
Exporting these routines to Routines/
```

We are then prompted to add recipients. These can be email addresses, or people in the NEW PERSON file. Once we have finished adding recipients, the Patch Module automatically generates an Patch message and sends it out.

Next, the Patch Module displays some version-control information, which

can be imported to commercial software such as Git. Version 2.5 of the Patch Module is the first one to support industry version-control software, and we suggest that all sites take advantage of this new feature.

Copy a Patch into a New Patch

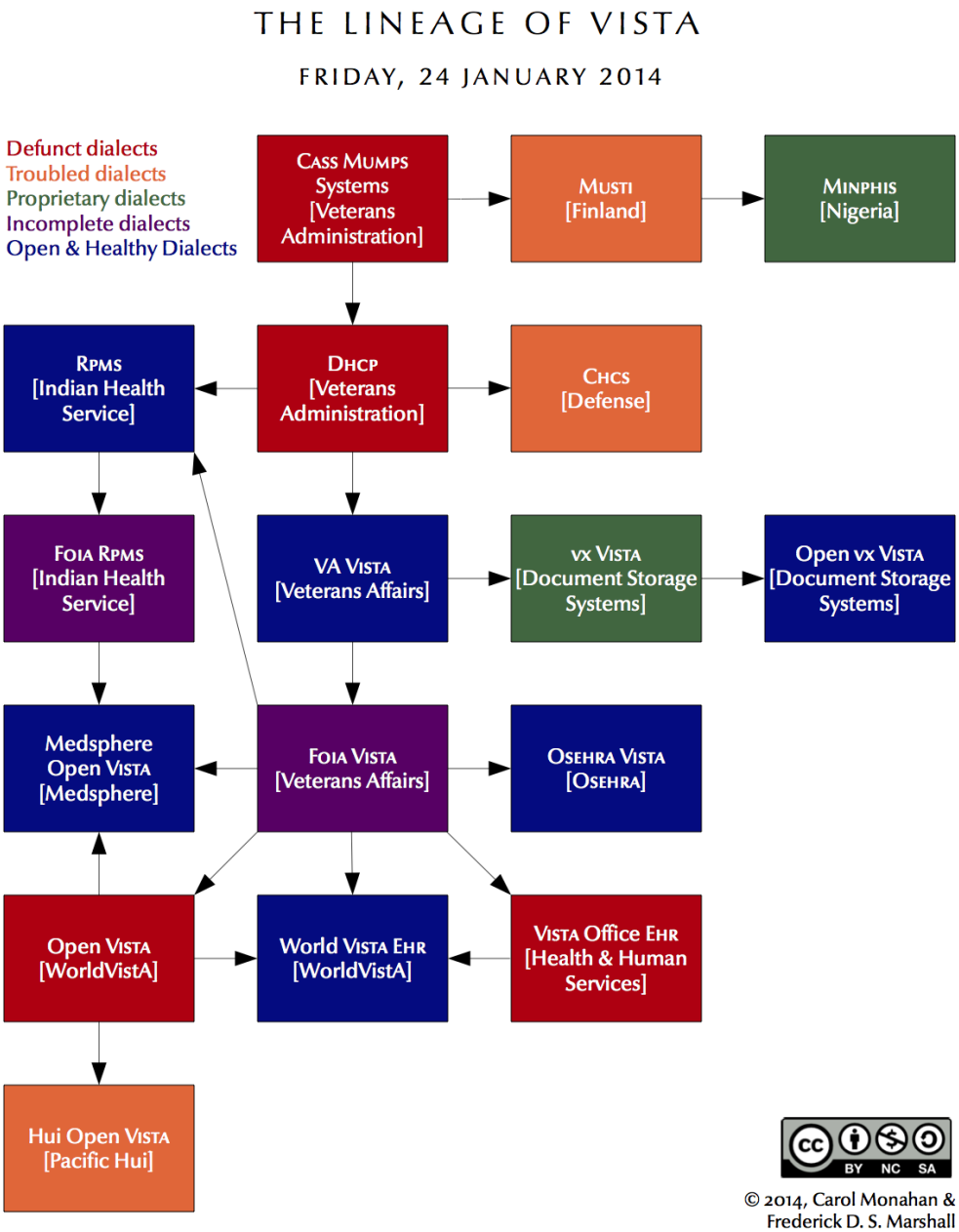
This function permits authorized developers of a package to copy information from an existing patch into a new patch. Generally, this option is not used in secondary development. Remember that if you are the one to initiate a patch—even a patch for a patch—then you are doing primary development. Please consult the Primary Developers User manual for more information.

Create a PackMan Message

This option allows a developer to create a Packman message from a local routine directory. Use this option to get a snapshot of what the routines currently look like on Forum. Since Forum is always kept patched and up-to-date, it is a good reference to use.

Secondary development is a complicated process. It often involves having multiple versions of the same patch installed or loaded in various environments. It is easy to lose track of where you are, and more importantly, of where the software is. Keep in mind that this resource is always available to you; you can always grab a snapshot of what a current system is "supposed" to look like, and work from there.

Appendix A: The Lineage of VistA


THE LINEAGE OF VISTA
FRIDAY, 24 JANUARY 2014
Defunct dialects
Troubled dialects
Proprietary dialects
Incomplete dialects
Open & Healthy Dialects
Cass Mumps Systems [Veterans Administration]
Musti [Finland]
Minphis [Nigeria]
Rpms [Indian Health Service]
Dhcp [Veterans Administration]
Chcs [Defense]
Foia Rpms [Indian Health Service]
VA Vista [Veterans Affairs]
vx Vista [Document Storage Systems]
Open vx Vista [Document Storage Systems]
Medsphere Open Vista [Medsphere]
Foia Vista [Veterans Affairs]
Osehra Vista [Osehra]
Open Vista [WorldVistA]
World Vista Ehr [WorldVistA]
Vista Office Ehr [Health & Human Services]
Hui Open Vista [Pacific Hui]
BY NC SA
© 2014, Carol Monahan & Frederick D. S. Marshall
Vista Expertise Network's The Lineage of Vista is licensed under the Creative Commons Attribution-Noncommercial-Share Alike 3.0 United States License (http://creativecommons.org/licenses/by-nc-sa/3.0/us/).

Glossary

Application	An administrative division of VistA that automates part or all of one hospital or clinical service. Pharmacy and Nursing are examples of applications.
Application Version	A complete new release of an application. Versions are numbered sequentially.
Build	See KIDS Build
Checksum	A number unique to any given version of a software element. Even a small change to the software will change the checksum, so checksums are used to detect changes and verify a particular version.
Dialect	See VistA Dialect
Distribution	See KIDS Distribution
FOIA	Freedom of Information Act. The term “FOIA” can refer to the Act itself, or to a request sent to the government under the auspices of the Act.
Forum	A VistA system used as the hub of an organization’s VistA software lifecycle. VA and OSEHRA each have their own Forums.
Gerrit	A code-review system for use with a repository such as Github.
Git	A version-control system.

Github	A platform that hosts repositories using the Git system.
Host-File Format	A file-based format for a KIDS distribution. It consists of two parts: the KIDS file and the text file.
KIDS	The Kernel Installation and Distribution System. KIDS is the primary method for preparing a patch for the Patch Module, as well as the mechanism for installing patches.
KIDS Build	The "manifest" of a KIDS distribution, which lists all the components included in the distribution.
KIDS Distribution	A host file or Packman message containing a software update and associated tools and conversions for applying it.
KIDS File	In a host-file format, the portion of the file containing the software.
KIDS Install	A record describing what happened during each installation of a KIDS distribution at a specific site.
Local Modification	A change to VistA made for a specific facility or organization. Local modifications are necessary in VistA, but result in changes to checksums that make version control more challenging.
Mailman	VistA's native email system. Patches can be distributed using Mailman's Packman module.
Namespace	A convention for naming VistA package elements. Each developer or organization is assigned a

	namespace, which is a unique character string, to be used use in naming routines, options, and other package elements. Namespacing helps keep similar elements from different developers distinct and easily identifiable.
Numberspace	Similar to a namespace, a numberspace is a unique numeric string assigned to a developer or organization. Numberspaces are used for VistA elements that have numbers rather than names.
Package	A distribution of a new version of an application.
Packman	A module of Mailman used to ship patches and other software.
Packman Format	A format for a KIDS distribution designed for use with Packman.
Packman Message	Any email message that contains a KIDS distribution in Packman format.
Patch	Any small change or update intended for installation in an active VistA system. Most patches can be installed with minimal disruption to the system or its users.
Patch Completer	The developer who reviews the patch developer's work, then updates the status of the patch in the Patch Module to "completed."
Patch Developer	Person who initially entered the information on the patch into the Patch Module. That person will be listed as the "developer" in the Patch Module, whether they did any actual development work or not.

Patch ID	A multi-part identification number for a patch, which includes the application namespace, the application version number, and the patch number.
Patch Message	An email message that contains a patch description and a Packman-format KIDS distribution. This is the default method for the Patch Module to distribute patches.
Patch Number	Unique number given to a patch, as it relates to the specific application and version. Patch numbers are numberspaced, so patches from different sources can be immediately distinguished.
Patch Reviewer	In secondary development, the developer who reviews the released patch to determine what kind of secondary development might be needed.
Patch Stream	The series of patches developed and released for a specific application or dialect.
Patch Subscriber	A person or organization who has signed up to receive a particular patch stream.
Patch Verifier	Specialist who confirms that the patch is functionally complete, and meets all standards. Verifiers make the decision to release the patch.
Primary Developer	The person or team who initiates the patching process and releases a new patch.
Primary Development	The actions involved in creating and distributing a new patch.

Repository	Online electronic storage which houses reference versions of a specific software. Generally, one version is designated as the "official" version. OSEHRA provides repositories for OSEHRA VistA and FOIA VistA.
Required Patch	A prerequisite patch. All patches should list their required patches—that is, their prerequisites—for installation.
Secondary Developer	The person or team who re-purposes a released patch for their VistA dialect.
Secondary Development	The actions involved in re-purposing a patch for a different VistA dialect.
Sequence Number	Unique number assigned when a patch is verified. It determines the default order in which patches should be installed.
Text File	In host-file format, the portion of the file that contains the patch description.
Version	See Application Version
Version Control	A system or methodology for ensuring that all software within a given organization is the same version.
Version Number	The sequential number of the current application version. Each VistA application has its own version number. For example, the current version of Laboratory is 5.2, while the current version of Fileman is 22.2.
VistA Dialect	A unique, stable version of VistA supported by a

	specific vendor. Popular VistA dialects include OSEHRA VistA, vxVistA (Document Storage Systems), Medsphere Open VistAand WorldVistA EHR.
VistA Service Pack	A bundle of VistA packages and patches, which can be used to upgrade an existing VistA system.
VistA Snapshot	A copy of an existing VistA system. A VistA snapshot is most commonly used to clone a new VistA instance.